notes from 5/13 meeting

* figure 6, define with define-language in redex
* figure 7, define with define-reduction-relation
* imperative language- main thing is assignment, if statements, and while loops
* program is series of statements
* skip means done evaluating one statement, can continue with next using store from evaluation of previous statements
* define judgement form for down arrow, relation between store (sigma), expr, value (result of evaluating expr), taint (of result)
  + down arrow used as side condition in rules for squiggly arrow (means after one step)
* bin op: binary operations, prof suggests using delta and just use add1 and mult
* suggest making taint numbers, and meet operator becomes max or min (depending on how taint is defined)
* delta, give it variable and it returns value
* tau, give it variable and it returns taint of variable
* alloc in figure 7 (what to do when given x=alloc for statement) – extends delta, means initialized x with taint tau, will need meta-function to lookup in store
* store consists of heap (map of vars to values), taint map Tau (map of vars to taints)
* don’t need evaluation contexts, no nesting because just sequence of statements
* (\Sigma, while e do S) --> (\Sigma, S; while e do S) for While-T rule (incorrect in paper)
* suggests that we write syntax to be more parenthesis friendly, don’t have to follow their syntax so tightly.